Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento Ingeniería Matemática

# Optimización No lineal

# Tarea 3

Integrantes: Bruno Hernandez
Benjamín Madariaga
Profesores: Jorge Amaya
Fecha 21 de julio de 2020
Santiago, Chile

# 1. Resumen

El siguiente reporte refleja la metodología y solución numérica de un problema de optimización de distancias planteado en el curso MA5701: Optimización No Lineal, semestre otoño 2020.

El problema plantea dos conjuntos de clientes perimetrados en un radio fijo, con el objetivo de encontrara los puntos óptimos que minimicen las distancias hacia todos los puntos del conjunto. La implementación numérica del problema se realizó en el lenguaje de programación `Python`, gracias a las librerías `NumPy` para el manejo de objetos numéricos y vectores, `matplotlib` para la generación de representaciones gráficas de los resultados y `SciPy` que suministra los algoritmos de optimización necesarios.

Mediante el desarrollo del planteamiento uno se ha de percatar de la no linealidad de la función objetivo del problema al tratarse de distancias euclideanas. Más adelante, al obtener los primeros resultados, notaremos la importancia en la simetría de los conjuntos de puntos objetivos para asegurar una condición de unicidad en las soluciones existentes, bajo este sentido, la distribución de los datos jugará un rol de suma importancia, más aún que la cantidad de puntos.

En la parte final se quitará una de las condiciones, generando otro grado de libertad en el cual los algoritmos podrán encontrar nuevos puntos críticos que antes se habrían descartado.

En la parte final de este reporte se planteará como propuesto buscar problemas que resulten equivalentes la ya planteado, con el fin de facilitar el trabajo computacional al momento de hallar una solución numérica. Acompañado a esto, se facilitarán los códigos para que, quien lo amerite, se pueda reproducir estos resultados.

# 2. Contexto

Se considera el plano de una parte circular ($S$) de la ciudad, de radio igual a 1000 metros. Suponga que cada 100 metros exactos se encuentra un cliente, el cual debe dirigirse a uno de dos puntos ($p_1$ y $p_2$) a retirar un producto. Cada cliente sabe exactamente a qué punto de entrega debe ir, no tiene elección. Los dos puntos de entrega ($p_1$ y $p_2$) están ubicados en el borde de la región circular y deben estar a distancia máxima uno del otro. Son puntos del plano, de coordenadas reales.

## a. Problema

La problemática de esta situación consiste en la decisión de *donde* localizar el par de puntos de entrega, de manera que la suma de las distancias recorridas por los clientes sea mínima.

## b. Modelamiento teórico

Definamos el conjunto de puntos que representen a las personas que se dirigen al punto $p_1 = (x_1, y_1)$ como $\mathcal{F}_1$, así mismo definamos el conjunto de puntos que representan a las personas que se dirigen al punto $p_2 = (x_2, y_2)$ como $\mathcal{F}_2$. Utilizando la distancia euclideana en $\mathbb{R}^2$, este caso se transforma en un problema de minimización con restricciones.

$$\begin{aligned} &\underset{x_1, y_1, x_2, y_2}{minimizar} && \sum_{(x,y)\in\mathcal{F}_1} \sqrt{(x_1-x)^2+(y_1-y)^2} + \sum_{(x,y)\in\mathcal{F}_2} \sqrt{(x_2-x)^2+(y_2-y)^2} \\ &sujeto\,a && x_1^2 + y_1^2 = x_2^2 + y_2^2 = r^2 \end{aligned} \tag{2.1}$$

Donde la condición corresponde a que los puntos se encuentren en la frontera de $S = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq r^2\}$, con $r = 1000$ para el caso particular de este problema.

Dado que los puntos de entrega deben cumplir con esta restricción, se considera la parametrización de los vectores de la forma

$$\forall (x, y) \in \partial S,\ \ p(x, y) = p(r, \theta) = r \cdot (cos(\theta), sin(\theta)), \quad \theta \in [0, 2\pi], \tag{2.2}$$

con lo que el problema pasa a ser un problema no lineal sin restricciones de dos variables $\theta_1, \theta_2$.

$$\underset{\theta_1, \theta_2 \in [0,2\pi]}{minimizar} \quad \sum_{\vec{a}\in\mathcal{F}_1} ||\vec{p}(\theta_1) - \vec{a}|| + \sum_{\vec{b}\in\mathcal{F}_2} ||\vec{p}(\theta_2) - \vec{b}|| \tag{2.3}$$

Sin embargo, dada la continuidad de la norma euclideana y el dominio compacto de $(\theta_1, \theta_2)$ ( $\in [0, 2\pi]^2$ ), deducimos que el problema 2.3 es un problema factible.

Estudiaremos dos versiones de este problema. Primero restringiremos el segundo punto de entrega a que sea diametralmente opuesto a al primero. Notamos que con esta restricción el segundo punto queda totalmente determinado por la posición del primero, por lo que se pierde una variable del problema.

$$\underset{\theta \in [0,2\pi]}{minimizar} \quad \sum_{\vec{a}\in\mathcal{F}_1} ||\vec{p}(\theta) - \vec{a}||^2 + \sum_{\vec{b}\in\mathcal{F}_2} ||\vec{p}(\theta + \pi) - \vec{b}||^2 \tag{2.4}$$

Luego analizaremos el problema sin esta restricción para el segundo punto, con lo que volvemos al planteamiento 2.3.

Para cada una de estas versiones del problema se resolverá el problema para 3 distribuciones distintas de $\mathcal{F}_1$ y $\mathcal{F}_2$. Estas consideran un total de clientes determinados por una grilla de puntos equiespaciados por una distancia horizontal y vertical de 100 metros al interior de $S$. Esto es, si

$$\mathcal{A} = \{100i \in \mathbb{R} : \ i = -10, \cdots, 10\},$$

entonces podemos escribir el conjunto total de clientes como

$$\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2 = (A \times A) \cap S.$$

Esto se visualiza de mejor manera en la Figura 2.1, donde el conjunto $\mathcal{F}$ corresponde a todos los puntos rojos que se encuentran al interior del círculo azúl ($S$).

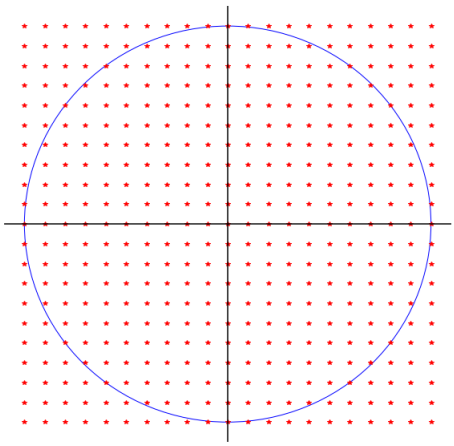

Figura 2.1: Descripción gráfica del conjunto $S$ y sus clientes.

En adelante a los conjuntos $\mathcal{F}_1$ y $\mathcal{F}_2$ restringidos a este total de clientes les llamaremos $S_1$ y $S_2$.

# 3. Resultados

## a. Primer caso

Consideremos el conjunto $S_1$ de todos los clientes ubicados en los puntos de coordenadas $(i, j)$, con $i = -500, \cdots, 500$ y $j = -500, \cdots, 500$, que deben ir al punto $p_1$. Los demás, $S_2$, van al punto $p_2$.

Notamos que esta distribución de clientes presenta simetría respecto de ambos ejes y en torno a las rectas $y = x$ e $y = -x$, poniendo en duda la unicidad de la solución del problema que resulta ser equivalente bajo rotaciones.

Se encontró una solución óptima en $\vec{p}_1 = (707.1, 707.1)$, esto implica que $\vec{p}_2 = (-707.1, -707.1)$. Esta solución corresponde a $\hat{\theta} = \pi/4$. Vemos que por la simetría del problema se encontraban distintas soluciones, pero que todas ellas eran equivalentes en cuanto al valor que lograban en la función objetivo. Estas corresponden a tomar los valores $\theta = k\pi/2 + \pi/4, k = 0, 1, 2, 3$.

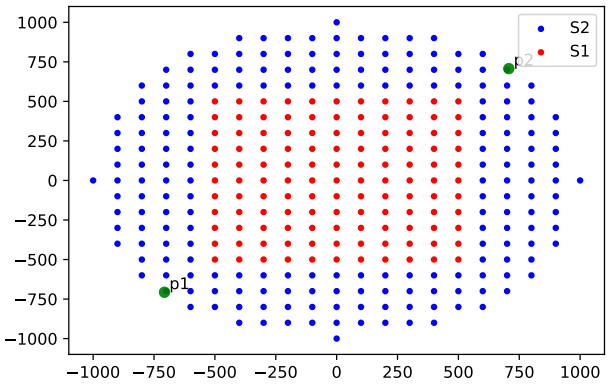


Figura 3.1: Solución para la primera distribución.

## b. Segundo caso

Consideremos el conjunto $S_1$ de todos los clientes ubicados en los puntos de coordenadas $(i, j)$, con $i = 0, \cdots, 500$ y $j = -500, \cdots, i$, que deben ir al punto $p_1$. Los demás, $S_2$, van al punto $p_2$. (ver Figura 3.2)

Se encontró que la solución óptima $\vec{p}_1 = (910.8, -412.9)$, esto implica que $\vec{p}_2 = (-910.8, 412.9)$. En este caso el problema no presentaba la simetría anterior, por lo que es una solución global.

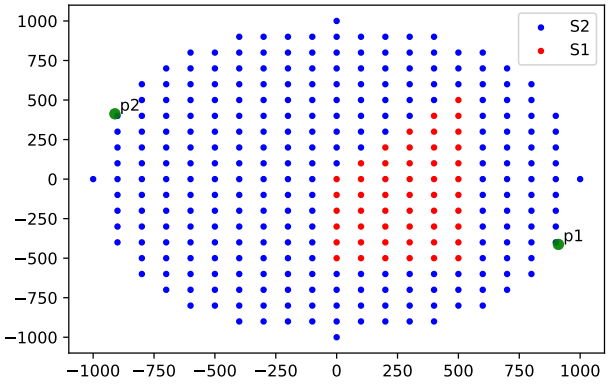


Figura 3.2: Solución para la segunda distribución.

Notamos que el punto óptimo se encuentra en el diámetro que ajusta su cercanía al centro de masa de los dos conjuntos, $S_1$ y $S_2$.

## c. Tercer caso

Consideremos el conjunto $S_1$ determinado por puntos específicos elegidos según un sorteo, según el cual se seleccionó $S_1 = \{$(0,0), (0,100), (0,300), (0,500), (100,200), (100,300), (200,-200), (-300,300), (1000,0), (-300,-400)$\}$ que deben ir al punto $p_1$. Los demás, $S_2$, van al punto $p_2$.

Se encontró que la solución óptima $\vec{p}_1 = (600, 800)$, esto implica que $\vec{p}_2 = (-600, -800)$. Nuevamente el problema no tiene simetría, por lo que también es una solución global.

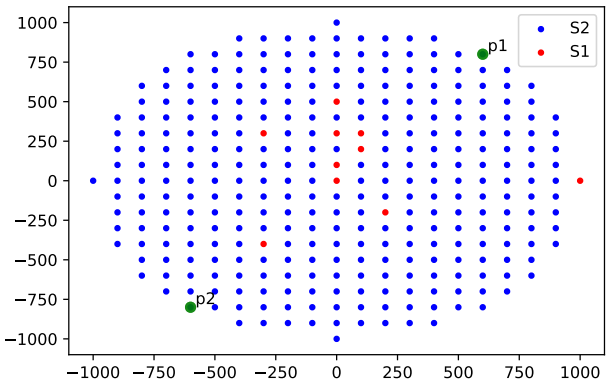


Figura 3.3: Solución para la tercera distribución.

## d. Sin restricción de diametralidad

Finalmente volvemos a considerar las 3 distribuciones distintas para $S_1$ y $S_2$, pero quitando la restricción de que el punto de entrega $p_2$ debe estar diametralmente opuesto a $p_1$.

En la tabla 3.1 vemos los resultados para los distintos casos. Es importante notar que por la simetría ya mencionada, en la primera configuración del problema existían múltiples resultados equivalentes. Además, al no existir la restricción de diametralidad, $\vec{p}_1$ y $\vec{p}_2$ podían incluso tomar valores iguales, ya que ambos se minimizaban en $\theta \in \{k\pi/2 + \pi/4, k = 0, 1, 2, 3\}$.

Para la segunda y tercera configuración de $S_1$ y $S_2$ podemos asegurar que los óptimos encontrados son globales. Además, notamos que son resultados similares a los encontrados para el problema restringido 2.4, donde los puntos óptimos están cercanos a cumplir la condición de estar a distancia máxima el uno del otro.

| Distribución | Variable | Valores |
|---|---|---|
| $1^{\text{st}}$ | $\vec{p}_1$ | (-707.1, 707.1) |
| | $\vec{p}_2$ | (-707.1, -707.1) |
| $2^{\text{nd}}$ | $\vec{p}_1$ | (908.5, -417.9) |
| | $\vec{p}_2$ | (-911.5, 411.4) |
| $3^{\text{rd}}$ | $\vec{p}_1$ | (389.4, 921.1) |
| | $\vec{p}_2$ | (-704.9, -709.3) |

Tabla 3.1: Resultados para el problema sin restricción 2.3.

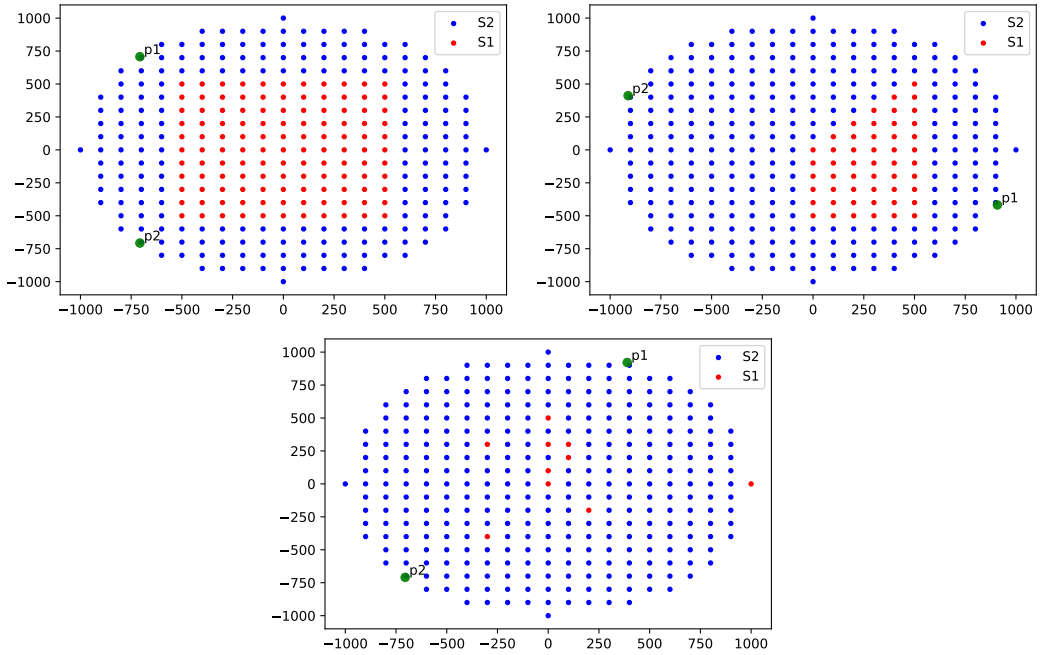


Figura 3.4: Solución para las tres distribuciones, eliminando la restricción de puntos de entrega a distancia máxima.

# 4. Conclusiones

El método de resolución planteado, mediante una parametrización, fue una pieza clave al momento del análisis de la unicidad de las soluciones, puesto que puso en juego el efecto de funciones trigonométricas que implican directamente la periodicidad de los valores óptimos. Sin embargo, no hay forma previa de notar el efecto de la distribución de puntos para la detección de simetrías en el planteamiento teórico del problema. Para este último punto, se propone que en futuros análisis del problema, se plantee la inclusión del centro de masa de los subconjuntos como un valor determinante en la solución, pues este entrega información relevante de la simetría de la nube de datos.

Además, se introduce como objetivo generar problemas equivalentes al expresado en el Problema 2.3 para mejorar la implementación numérica de un algoritmo de optimización y perder la dependencia entre el punto inicial del algoritmo y el resultado obtenido. Una alternativa podría ser el *problema dual-lagrangiano*.

En las siguientes páginas se disponen los códigos utilizados en el lenguaje `Python`, para la reproducibilidad de los resultados.

# Tarea 3 ONL

July 21, 2020

## 1 5. Código

[19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.spatial import distance
```

[ ]:
```
def get_distances(S,p):
    dist=distance.cdist(S,np.array(p).reshape((1,2)))
    return sum(dist)

def get_p(theta):
    return (np.cos(theta)*1000,np.sin(theta)*1000)
```

[ ]:
```
def solve_modelo(dist_s1,p2_max_dist,N=20,plot=False):
    """
    Función principal para la resolución del problema.
    Soluciona la minimización de suma de distancias de dos conjuntos de puntos␣
 ↪inscritos en la circunferencia de centro 0
    y radio 1000 a dos puntos de entrega.

    Parametros:
        dist_s1: Int. Distribución inicial de puntos del conjunto S1. Toma␣
 ↪valores en {1,2,3}
            1: Distribución de puntos (i,j) con i = 500, . . . , 500, j = 500, .␣
 ↪. . , 500.
            2: Distribucion de puntos (i,j) con i = 0, . . . , 500, j = 500, . .␣
 ↪. , i.
            3: Distribución de puntos seleccionados aleatoriamente descritos en␣
 ↪el problema.

        p2_max_dist: Bool. True si el segundo punto de entrega se encuentra␣
 ↪diametralmente opuesto a p1.
```

```
                    Si es False, este segundo punto es tambiénparte de la␣
↪optimización.

        N: Int. Cantidad de puntos iniciales distintos a ocupar. Son elegidos␣
↪aleatoriamente según una dist uniforme a lo largo de la circunferencia.

        plot: Bool.
    """
    S = [((i*100),(j*100)) for i in range(-10,10+1) for j in range(-10,10+1) if␣
↪(i*100)**2+(j*100)**2<=(1000)**2]
    if dist_s1==1:
        S1 = [((i*100),(j*100)) for i in range(-5,5+1) for j in range(-5,5+1)]
    elif dist_s1==2:
        S1 = [((i*100),(j*100)) for i in range(0,5+1) for j in range(-5,5+1) if␣
↪j<=i]
    elif dist_s1==3:
        S1= [(0, 0), (0, 100), (0, 300), (0, 500), (100, 200), (100,␣
↪300),(200,-200), (-300, 300), (1000, 0), (-300,-400)]
    else:
        raise ValueError('Valor dist_s1 no reconocido. Los valores aceptados son␣
↪{1,2,3}.')

    S2 = list(set(S).difference(set(S1)))

    F = lambda theta: get_distances(S1,get_p(theta[0])) +␣
↪get_distances(S2,get_p(theta[1]))

    if p2_max_dist:
        f = lambda theta: F([theta,theta+np.pi])

        lb = [0]
        ub = [2*np.pi]
        cotas = list(zip(lb,ub))

        best_x=[0,0]
        best_val=np.infty

        for i in range(N):
            p0 = np.random.rand(1)*2*np.pi

            m = minimize(f,p0, options={'maxiter':1000}, bounds = cotas)
            if m.fun < best_val:
                best_x=m.x
                best_val=m.fun
        p = [get_p(best_x),get_p(best_x+np.pi)]
    else:
```

```
        f = lambda theta: F(theta)

        lb = [0,0]
        ub = [2*np.pi,2*np.pi]
        cotas = list(zip(lb,ub))

        best_x=[0,0]
        best_val=np.infty

        for i in range(N):
            p0 = np.random.rand(2)*2*np.pi

            m = minimize(f,p0, options={'maxiter':1000}, bounds = cotas)
            if m.fun < best_val:
                best_x=m.x
                best_val=m.fun
        p = [get_p(best_x[0]),get_p(best_x[1])]

    if plot:
        p_label=['p1','p2']
        plt.scatter(list(zip(*S2))[0],list(zip(*S2))[1],c='b',marker='.
↪',label='S2')
        plt.scatter(list(zip(*S1))[0],list(zip(*S1))[1],c='r',marker='.
↪',label='S1')
        plt.scatter(list(zip(*p))[0],list(zip(*p))[1],c='g',marker='o',alpha=0.7)
        plt.scatter(list(zip(*p))[0],list(zip(*p))[1],c='g',marker='o',alpha=0.7)
        plt.legend()
        for i, txt in enumerate(p_label):
            plt.annotate(txt, p[i], xytext=[a+20 for a in p[i]])
            plt.savefig('sol_dist_{}{}.pdf'.
↪format(dist_s1,'_maxdist'*p2_max_dist),bbox_inches='tight')
    return p,m
```

[ ]:
```
p,m=solve_modelo(1,True,plot=True)
```